PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-055092
(43)Date of publication of application : 27.02.1996

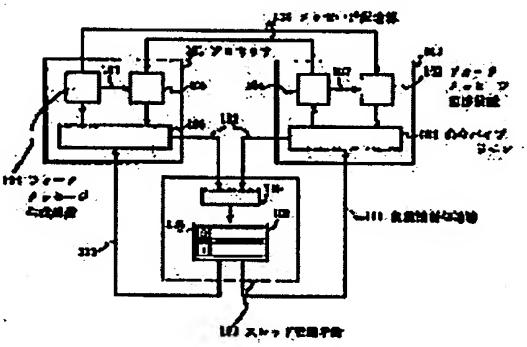
(51)Int.Cl. G06F 15/16
G06F 9/38
G06F 9/46

(21)Application number : 06-188147
(22)Date of filing : 10.08.1994

(71)Applicant : NEC CORP
(72)Inventor : MOTOMURA MASATO
TORII ATSUSHI

(54) PROCESSOR SYSTEM AND ITS CONTROL METHOD

(57)Abstract:
PURPOSE: To provide a dynamic parallelism control mechanism and its control method which extract the excessive parallelism to prevent the degradation of performance with respect to the processor system which makes the thread fork to another processor by a fork instruction to perform the parallel processing.
CONSTITUTION: A thread management means 121 is provided besides plural processors 101 to realize the processor system. The thread management means 121 is provided with a load state table 109, and this table 109 consists of plural load state counters 108 which show the load states of processors 101. Load state counters 108 are accessed to find the dynamic load states of processors 101, and they are used to control the parallelism.



LEGAL STATUS

[Date of request for examination]	10.08.1994
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	2684993
[Date of registration]	15.08.1997
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	

(19)日本国特許庁（ＪＰ）　(12)特　許　公　報（Ｂ２）　(11)特許番号

第2684993号

(45)発行日　平成９年(1997)12月３日　(24)登録日　平成９年(1997)８月15日

(51)Int.Cl.⁶		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/16	３８０		Ｇ０６Ｆ	15/16	３８０Ｚ
	9/38	３１０			9/38	３１０Ｊ
	9/46	３４０			9/46	３４０Ｄ

請求項の数９(全 ９ 頁)

(21)出願番号　特願平6－188147

(22)出願日　平成６年(1994)８月10日

(65)公開番号　特開平8－55092

(43)公開日　平成８年(1996)２月27日

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　本村　真人
東京都港区芝五丁目７番１号　日本電気株式会社内

(72)発明者　島居　淳
東京都港区芝五丁目７番１号　日本電気株式会社内

(74)代理人　弁理士　京本　直樹　(外２名)

審査官　石井　茂和

(56)参考文献　特開　平３－40034（ＪＰ，Ａ）
特開　平５－313922（ＪＰ，Ａ）
特開　昭58－66633（ＪＰ，Ａ）

(54)【発明の名称】　プロセッサシステムとその制御方法

1

(57)【特許請求の範囲】

【請求項1】複数の命令から構成されるスレッドを外部へフォークするスレッドフォーク手段と、外部からフォークされた前記スレッドを発行するスレッド発行手段と、発行された前記スレッド内の複数の前記命令を順次処理する命令パイプラインとを備え、スレッドのフォークを指示するフォーク命令を制御方法として備えたプロセッサを複数個用い、ある前記プロセッサにおいて前記フォーク命令を実行することにより前記スレッドフォーク手段を介して前記スレッドをフォークし、別の前記プロセッサにおいてフォークされた前記スレッドを前記スレッド発行手段を介して発行して前記スレッドの複数の前記命令を前記命令パイプラインで実行することにより並列処理を実現するプロセッサシステムであって、これら複数個の前記プロセッサ間でスレッド管理手段を共有し、前記スレッド管理手段は前記プロセッサシステムの負荷状態を示す負荷状態カウンタを保有し、しかもプロセッサの命令セットの一部としてスレッドの数を制限する命令を組み込み、この命令を用いて前記スレッド管理手段を駆動することを特徴とするプロセッサシステム。

【請求項2】プロセッサの命令セットの一部としてスレッドの数を制限する命令として、少なくともカウンタロック命令、カウンタアンロック命令を有する請求項1記載のプロセッサシステム。

【請求項3】請求項1に記載のプロセッサシステムの制御方法であって、負荷状態カウンタロック命令を規定し、ある前記プロセッサが前記負荷状態カウンタロック命令を実行することにより、当該の前記プロセッサが前記スレッド管理手段内の前記負荷状態カウンタをロックし、他の前記プロセッサが前記負荷状態カウンタの値を

2

3

変更できないようにすることを特徴とするプロセッサシステムの制御方法。

【請求項4】請求項1に記載のプロセッサシステムの制御方法であって、負荷状態カウンタアンロック命令を規定し、ある前記プロセッサが前記負荷状態カウンタアンロック命令を実行することにより、当該の前記プロセッサが前記スレッド管理手段内の前記負荷状態カウンタをアンロックし、他の前記プロセッサが前記負荷状態カウンタの値を変更できるようにすることを特徴とするプロセッサシステムの制御方法。

【請求項5】請求項1に記載のプロセッサシステムの制御方法であって、前記フォーク命令を実行することにより前記プロセッサが前記スレッドフォーク手段を介して前記スレッドをフォークする際に、前記負荷状態カウンタをアンロックすることを特徴とするプロセッサシステムの制御方法。

【請求項6】請求項1、3、4または5に記載のプロセッサシステムの制御方法であって、前記フォーク命令を実行することにより前記プロセッサが前記スレッドフォーク手段を介して前記スレッドをフォークする際に前記負荷状態カウンタの値を増やすことと、ある前記プロセッサが前記スレッドの実行を終了する際に前記負荷状態カウンタを減じることを特徴とするプロセッサシステムの制御方法。

【請求項7】請求項1、3、4、5または6に記載のプロセッサシステムの制御方法であって、前記フォーク命令を実行することにより前記プロセッサが前記スレッドフォーク手段を介して前記スレッドをフォークする際に、前記スレッド管理手段を排他的に参照し、前記負荷状態カウンタの値が一定の負荷状態以下を示しているかどうかを条件として前記スレッドのフォークが可能かどうかを判定し、フォークが可能な場合は前記スレッドのフォークを行ない、フォークが不可能な場合は当該する前記プロセッサ上で前記スレッドを逐次的に実行することを特徴とするプロセッサシステムの制御方法。

【請求項8】請求項1、3、4、5、6または7に記載のプロセッサシステムの制御方法であって、前記フォーク命令を実行することにより前記プロセッサがスレッドをフォークする際に、フォークされた前記スレッドが前記プロセッサシステムを構成する複数の前記プロセッサのうちどの前記プロセッサで実行されるかを、前記負荷状態カウンタの値を参照することにより、前記スレッド管理手段が決定することを特徴とするプロセッサシステムの制御方法。

【請求項9】請求項1、3、4、5、6、7または8に記載のプロセッサシステムの制御方法であって、スレッド確保命令を規定し、前記スレッド確保命令を実行することにより、前記プロセッサが前記スレッド管理手段を参照して前記スレッドをフォークすることが可能かどうかを確認し、可能である場合は前記スレッド管理手段内

4

の前記負荷状態カウンタの値を増やすことを特徴とするプロセッサシステムの制御方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は情報処理システムの中核をなすプロセッサに関し、更に複数のプロセッサ間でスレッドをフォークしながらプログラムの並列処理を実行するプロセッサシステムに関する。

【0002】

【従来の技術】一般に、プロセッサは、命令列で構成されるプログラムから一つの命令を読み込み、その命令を解釈して指示された処理を実行する、という手順を繰り返すことにより、プログラムが意図するところの全体処理を実現する。この際、プログラムの実行に必要なデータは適宜、命令の指示に従いながらメモリ装置から読み込まれ、あるいは実行結果がメモリ装置に書き込まれる。従来技術としてのプロセッサの基本的な動作は、このようにプログラム内に記述された命令の指示に従って内部的な処理を順次行なっていくものであり、このような処理を担当する部分を命令パイプラインと総称する。

【0003】一方、このようなプロセッサを使用してプロセッサシステムを構成するにあたっては、上記のような命令パイプラインによる内部的な動作の他に、プロセッサの外部に接続された機器の制御を行なうため、あるいは外部の機器からの制御を受け入れるために外部とプロセッサ間のインタフェースを設ける必要がある。プロセッサから外部機器の制御を行なう従来技術としては、メモリマップドI／O方式や、特殊命令による制御方式が知られている。一方、外部からプロセッサの制御を行なう従来技術としては、割り込み処理技術が知られている。これらの従来技術についてはコンピュータ科学の教科書、例えば“Computer Organization & Design: The Hardware Software Interface”, David Patterson and John Hennessy, Morgan Kaufmann Publishers Inc. p. 566などに詳しく記載されている。

【0004】一般に、プロセッサに対して外部より割り込みが通知されると、ユーザープログラムから割り込み処理用のプログラム（割り込みハンドラと呼ばれる）へと当該プロセッサ上の実行プログラムが切り替わり、割り込みハンドラが通知された割り込みに対応した処理を行ない、元のユーザープログラムに戻るかまたは必要に応じて更に別のプログラムを起動する、といった手順が実行される。このように実行するプログラムが切り替わるため、これにともなってプログラムの実行環境自体も切り替わる。ここでプログラムの実行環境とはレジスタファイルに記憶されているデータやプログラムカウンタ、あるいはプロセスコントロールブロックの中身など

5

であり、切り替わり時には、後ほど再度使用するために、これらの実行環境をメモリ装置内に退避する必要がある。このようなオーバーヘッドが存在するため、割り込み処理は一般には長大な時間を有することが知られている。このような問題については、“The Interaction of Architecture and Operating System Design”, Proceedings of Fourth International Conference on Architectual Support for Programming Languages and Operating Systems, pp. 108−120, Thomas E. Anderson, Henry M. Levy, Brian N. Bershadand Edward D. Lazowska などに詳しく記載されている。

【０００５】一方、複数のプロセッサを用いて並列処理を実現するプロセッサシステムにおいては、プロセッサ間の通信が頻繁に行なわれる。これは処理の分担を図るための通信や同期をとるための通信が多発するためである。従来のプロセッサを使用したプロセッサシステムでは上記のプロセッサ間通信を割り込み処理技術を用いて実現していた。このため、プロセッサ間通信処理に大きなオーバーヘッドが存在し、これが並列処理により性能を向上する上でのボトルネックとなってしまっていた。

【０００６】このような問題を解決するため、従来技術の一つとしてマルチスレッドアーキテクチャと呼ばれるプロセッサのアーキテクチャが提案されている。ここでスレッドとは複数の命令から構成される命令列であり、一つのプログラムは複数のスレッドの集合として定義されるものとする。マルチスレッドアーキテクチャにおいては、スレッド単位で処理を複数のプロセッサに分割し、これらのスレッドを並列に処理する。このために、マルチスレッドアーキテクチャは、一般に、あるプロセッサ上で実行されているスレッドが別のプロセッサ上に新たにスレッドを生成するための機構及び命令、更にはあるプロセッサ上で実行されているスレッドと別のプロセッサ上で実行されているスレッドとの間で同期をとるための機構及び命令などを特徴として備えている。本発明はスレッド生成に関する新規技術を提案するものであるので、以下ではマルチスレッドアーキテクチャの技術について説明する。

【０００７】図8に従来技術のマルチスレッドアーキテクチャに基づくプロセッサシステムの構成例を示す。図8においてプロセッサシステムは複数のプロセッサ８１から構成されており、それぞれのプロセッサ８１は命令パイプライン８５とフォークメッセージ生成装置８４、フォークメッセージ記憶装置８３から構成される。

【０００８】ここで他のプロセッサ８１上に新たなスレッドを生成することを「スレッドをフォークする」と呼ぶことにする。各プロセッサ８１のアーキテクチャはフォーク命令をその命令セットの中に備えている。ここでフォーク命令とは、スレッドのフォークを指示する命令であり、この命令をあるプロセッサ８１の命令パイプライン８５で実行することによりスレッドがフォークされる。一般に、フォーク命令は、（１）どのプロセッサ８１にスレッドをフォークするか、（２）どのようなスレッドをフォークするか、また（３）どのようなデータに対する処理をフォークするか、などに関する指示を引数として与えられる命令である。一般に（１）の指示はプロセッサの番号をダイレクトに与えるか、あるいは別の引数の値を解釈するなどの手段により与えられる。（２）の指示はスレッドの先頭命令アドレス、（３）の指示は当該スレッドが使用するスタック領域の先頭アドレスとして与えられることが多い。

【０００９】あるプロセッサ８１上でフォーク命令が実行されると、フォークメッセージがフォーク先として指定されたプロセッサ８１にメッセージ伝達線８６を介して送られ、当該プロセッサのフォークメッセージ記憶装置８３に受信され、格納される。フォークメッセージはフォーク命令の引数として与えられた情報を含むメッセージである。フォークメッセージを受信したプロセッサ８１は、フォークメッセージ記憶装置８３でフォークメッセージを解釈し、命令パイプライン８５を用いて指定されたスレッドの実行を開始する。一般に、フォークメッセージを受信したプロセッサ８１が他のスレッドを実行中の場合が考えられる。このような場合に対応するため、フォークメッセージ記憶装置８３は複数のフォークメッセージを記憶できる機構を持つ必要がある。すなわち、プロセッサ８１上であるスレッドの実行が終了すると、スレッドメッセージ記憶装置８３に記憶された複数のフォークメッセージの中から一つをあるスケジュール規則に従って選び、選ばれたフォークメッセージにより指定されたスレッドの実行を命令パイプライン８５を用いて開始する。ここでスケジュール規則とは、単純なファーストインファーストアウトやラストインファーストアウトに基づくものなどや、プライオリティレベルなどによるものなどがある。

【００１０】以上簡単に説明したマルチスレッドアーキテクチャの代表的な例としては、“＊T. A Multi−threaded Massively Parallel Architecture”, Proceedings of 19thInternational Symposium on Computer Architecture, R. S. Nikhil, G. M. Papadopoulos and Arvind, pp. 156−167で発表されているアーキテクチャが挙げられる。なお、本アーキテクチャにおいては、上記フォーク命令をスタート命令と呼んでいる。

【００１１】以上説明したようなマルチスレッドアーキ

7

テクチャは、複数のプロセッサ間で並列処理を行なう際に、プロセッサ間通信のオーバーヘッドを大幅に削減することが出来る。このためより粒度の細かい単位で処理を分割し、プログラムに存在する細かい粒度の並列性を有効に利用することが出来る、という利点がある。

【００１２】

【発明が解決しようとする課題】以上従来の割り込み処理技術と比べた場合のマルチスレッドアーキテクチャ技術の利点を説明したが、この技術にも大きな欠点がある。それは並列性の制御が困難であるという点である。

【００１３】図8から明らかなように、従来技術のマルチスレッドアーキテクチャではあるプロセッサ８１がスレッドをフォークすると、受信側のプロセッサ８１は対応するフォークメッセージを受信し、格納する以外の手順をとることが出来ない。すなわち送信側のプロセッサ８１は一方的にフォークメッセージを送ることができ、一方受信側のプロセッサ８１は否応なくこれを受信しなければならない。例えば繰り返し数が巨大なループの各イテレーションをスレッドとしてフォークするような場合などから明らかなように、このようなアーキテクチャ上の制約は不必要な並列性を抽出してしまうという状況を招き、以下のような点で性能が低下してしまうことになる。

【００１４】・フォークメッセージ記憶装置８１はある一定数のフォークメッセージしか記憶できない。上記のような原因によりこの数を越えるフォークメッセージを受信することを余儀なくされた場合、割り込み処理などにより、フォークメッセージ記憶装置８１内に記憶されたフォークメッセージをメインメモリに退避する必要が生じる。これは大きなオーバーヘッドとなる。

【００１５】・大量のスレッドがフォークされた場合、多くのスレッドはかなり後になるまで実行されることがない。よって、これらのスレッドと同期をとる必要があるスレッドは長い時間同期待ちをしてしまうことになる。これによりプロセッサの有効利用率が低下し、処理性能が低下してしまう。

【００１６】なお上では説明を割愛したが、従来技術のままでも、フォークをしても性能の向上が見込めない場合、割り込み処理によりフォークをキャンセルするなどの方法により送信側が一方的にフォークするという状況を改めることが出来る。しかし、このような方法では割り込み処理によるオーバーヘッドが大きいため、いずれにしろ性能の低下は避けることが出来ない。

【００１７】以上説明したように、従来技術のマルチスレッドアーキテクチャに基づくプロセッサを用いたプロセッサシステムにおいては、並列性を制御することが困難なため、プロセッサシステムの負荷状況によっては、それぞれのプロセッサを有効利用するために必要とされる以上にスレッドをフォークしてしまうことにより、却って性能低下を招くという問題が生じる。

8

【００１８】本発明の目的は、動的な並列性制御手段とその制御方法を提案することによってこれらの問題を解決し、スレッドを利用した並列処理の実行を効率化するプロセッサ及びプロセッサシステムを提供することにある。

【００１９】

【課題を解決するための手段】本発明によるプロセッサ及びプロセッサシステムは、複数の命令から構成されるスレッドを外部へフォークするスレッドフォーク手段と、外部からフォークされた前記スレッドを発行するスレッド発行手段と、発行された前記スレッド内の複数の前記命令を順次処理する命令パイプラインとを備え、スレッドのフォークを指示するフォーク命令を制御方法として備えたプロセッサを複数個用い、ある前記プロセッサにおいて前記フォーク命令を実行することにより前記スレッドフォーク手段を介して前記スレッドをフォークし、別の前記プロセッサにおいてフォークされた前記スレッドを前記スレッド発行手段を介して発行して前記スレッドの複数の前記命令を前記命令パイプラインで実行することにより並列処理を実現するプロセッサシステムであって、これら複数個の前記プロセッサ間でスレッド管理手段を共鳴し、前記スレッド管理手段は前記プロセッサシステムの負荷状態を示す負荷状態カウンタを保有することを特徴とするプロセッサシステムとして構成される。

【００２０】本発明のプロセッサ及びプロセッサシステムにおいて、本発明の第1のプロセッサの制御方法は負荷状態カウンタロック命令を規定し、ある前記プロセッサが前記負荷状態カウンタロック命令を実行することにより、当該の前記プロセッサが前記スレッド管理手段内の前記負荷状態カウンタをロックし、他の前記プロセッサが前記負荷状態カウンタの値を変更できないようにすることを特徴とする前記プロセッサの制御方法を用い、また負荷状態カウンタアンロック命令を規定し、ある前記プロセッサが前記負荷状態カウンタアンロック命令を実行することにより、当該の前記プロセッサが前記スレッド管理手段内の前記負荷状態カウンタをアンロックし、他の前記プロセッサが前記負荷状態カウンタの値を変更できるようにすることを特徴とする前記プロセッサの制御方法を用い、前記フォーク命令を実行することにより前記プロセッサが前記スレッドフォーク手段を介して前記スレッドをフォークする際に、前記負荷状態カウンタをアンロックすることを特徴とする前記プロセッサの制御方法を用いる。

【００２１】本発明のプロセッサ及びプロセッサシステムにおいて、本発明の第2のプロセッサの制御方法は前記フォーク命令を実行することにより前記プロセッサが前記スレッドフォーク手段を介して前記スレッドをフォークする際に前記負荷状態カウンタの値を増やすことと、ある前記プロセッサが前記スレッドの実行を終了す

る際に前記負荷状態カウンタを減じることを特徴とする前記プロセッサシステムの制御方法を用いる。

【００２２】本発明のプロセッサ及びプロセッサシステムにおいて、本発明の第3のプロセッサの制御方法は前記フォーク命令を実行することにより前記プロセッサが前記スレッドフォーク手段を介して前記スレッドをフォークする際に、前記スレッド管理手段を排他的に参照し、前記負荷状態カウンタの値が一定の負荷状態以下を示しているかどうかを条件として前記スレッドのフォークが可能かどうかを判定し、フォークが可能な場合は前記スレッドのフォークを行ない、フォークが不可能な場合は当該する前記プロセッサ上で前記スレッドを逐次的に実行することを特徴とする前記プロセッサの制御方法を用いる。

【００２３】本発明のプロセッサ及びプロセッサシステムにおいて、本発明の第4のプロセッサの制御方法は前記フォーク命令を実行することにより前記プロセッサがスレッドをフォークする際に、フォークされた前記スレッドが前記プロセッサシステムを構成する複数の前記プロセッサのうちどの前記プロセッサで実行されるかを、前記負荷状態カウンタの値を参照することにより、前記スレッド管理手段が決定することを特徴とする前記プロセッサの制御方法を用いる。

【００２４】本発明のプロセッサ及びプロセッサシステムにおいて、本発明の第5のプロセッサの制御方法はスレッド確保命令を規定し、前記スレッド確保命令を実行することにより、前記プロセッサが前記スレッド管理手段を参照して前記スレッドをフォークすることが可能かどうかを確認し、可能である場合は前記スレッド管理手段内の前記負荷状態カウンタの値を増やすことを特徴とする前記プロセッサの制御方法を用いる。

【００２５】

【実施例】図1は本発明によるプロセッサシステムの第１の実施例を示すブロック構成図である。図1において、プロセッサシステムは複数のプロセッサ１０１から構成されており、プロセッサ１０１は命令パイプライン１０５とフォークメッセージ生成装置１０４、フォークメッセージ記憶装置１０３から構成される。フォークメッセージ生成装置１０４とフォークメッセージ記憶装置１０３は、メッセージ伝達線１０６によって接続されている。自プロセッサ内のフォークメッセージ生成装置１０４とフォークメッセージ記憶装置１０３はメッセージ伝達線１０７により接続されている。

【００２６】はじめに本実施例においてスレッド生成を行なう場合の基本的な動作について説明する。プロセッサ１０１の命令パイプライン１０５が、スレッド生成のフォーク命令を実行すると、フォークメッセージ生成装置１０４によってフォークメッセージが生成され、メッセージ伝達線１０６を介して、フォークメッセージ内で指定されたプロセッサ１０１のフォークメッセージキュー記憶装置１０３に送られる。プロセッサ１０１は現在実行しているスレッドの実行が終了もしくは中断した時に、フォークメッセージ記憶装置１０３からフォークメッセージを取り出し、それに従って新たなスレッドの実行を開始する。

【００２７】次に、本実施例におけるスレッド管理手段１２１について説明する。スレッド管理手段１２１はプロセッサ毎の負荷情報カウンタ１０８から構成される負荷状態テーブル１０９と解読装置１１０から構成される。スレッド管理手段１２１は各プロセッサのスレッドの実行情報を実行情報伝達線１１２によって集めることによって、該当するプロセッサの負荷状態カウンタ１０８の値を更新する。全プロセッサの負荷情報は、負荷状態テーブル１０９を各プロセッサ１０１が負荷情報伝達線１１１を介して読み出すことによって調べることが可能である。プロセッサ１０１はこの情報を用いて、負荷の軽い他のプロセッサ１０１に対してスレッド生成を要求するフォーク命令を実行することにより、負荷分散を行なうことが可能である。

【００２８】なお、本実施例ではプロセッサ2個の並列システムについて図示したが、プロセッサ数は2個に限定するものではなく、これ以上のプロセッサがシステム中に存在する場合も同様である。

【００２９】図2は、図1に示したプロセッサシステムの第1の実施例におけるスレッド管理手段についてその第2の実施例を示したブロック構成図である。本実施例は請求項2～4に対応するものである。図1においてスレッド管理手段２２１は第1の実施例におけるスレッド管理手段１２１に、負荷状態テーブル１０９の更新をロックするためのロック装置２０１を追加したものである。このロック装置２０１は、実行情報伝達線１１２を介した各プロセッサ１０１からの要求によって、負荷状態テーブル１０９の更新禁止と更新許可の設定を行なうことができる。更新禁止状態にある時は、更新禁止を要求した以外のプロセッサ１０１から負荷状態テーブル１０９を更新することはできなくなる。

【００３０】請求項2では、プロセッサ１０１の命令にカウンタロック命令を加え、この命令を実行した場合には実行情報伝達線１１２にロック要求を出す。解読装置１１０はロック要求を解読した場合には、ロック装置２０１が既にロック状態にない場合には、ロック状態にセットするとともにＩＤ番号を記憶する。ただしＩＤ番号は、プロセッサ番号や命令によって指定した番号とする。既にロック状態にある場合には、このロック命令実行が失敗したという情報をロック要求を行なったプロセッサ１０１に返す。

【００３１】請求項3では、プロセッサ１０１の命令にカウンタアンロック命令を加え、この命令を実行した場合には実行情報伝達線１１２にロック解除要求を出す。解読装置１１０はロック解除命令を解読した場合には、

11

ロック装置２０１が既にロック状態にあり、ロックをかけたプロセッサとロック解除要求を行なったＩＤ番号が同一である場合には、ロック状態を解除する。これ以外の場合にはロック解除命令実行が失敗したという情報をロック解除要求を行なったプロセッサ１０１に返す。

【００３２】請求項4では、プロセッサ１０１でフォークを行なう場合に、フォークの必要動作と請求項3のアンロック命令を同時に行なうフォーク命令を定義する。これにより、フォークを行なった時には、ロック解除命令を実行することなく、負荷状態テーブル１０９のロックを解除することが可能になる。

【００３３】本実施例で説明したような機能を持つカウンタロック命令、カウンタアンロック命令、フォーク命令を用いることによって以下のような動作が可能になる。まず、カウンタロック命令を実行して負荷状態カウンタ１０９をロックし、その値を読み出す。この値を用いてフォークによってスレッド生成を行なうプロセッサを決定してフォークを行なう。このことにより負荷分散とデータ配置を考えた並列処理が可能となる。フォークを行なう場合には、フォーク時に自動的に負荷状態カウンタ１０９のロックが解除されるためアンロック命令を実行する必要はない。モニタの値によってはフォークを行なわない方が得策である場合も考えられるが、この場合にはアンロック命令を実行して、フォークを行なわないで逐次処理を行なう。

【００３４】図3は、図1に示したプロセッサシステムの第1の実施例におけるスレッド管理手段についてその第3の実施例を示したブロック構成図である。本実施例は請求項5によるプロセッサシステムの制御方法に対応するものである。本実施例において、スレッド管理手段３２１は図1に示したスレッド管理手段１２１に、加算器３０１、減算器３０２、選択装置３０３を追加したものである。

【００３５】本実施例のスレッド管理手段３２１は、スレッドの実行開始実行を実行情報伝達線１１２によって受けとる。この情報は解読装置１１０によって解読され、そのプロセッサ１０１に対応する負荷情報カウンタ１０８の値を選ぶように選択装置３０３に指令を与える。加算器３０１によって、負荷状態カウンタ１０８の値に実行が開始されたスレッド数を加え、その値を負荷状態カウンタ１０８の新しい値とする。同様に、スレッド実行終了情報が伝えられた場合には、減算器３０２を用いて該当するプロセッサの負荷情報カウンタ１０８を減ずる処理を行なう。

【００３６】負荷情報テーブル１０９の値は、負荷情報伝達線１１１を介して各々のプロセッサが参照することが可能である。これらの手段によって、システム中の全てのプロセッサの負荷情報、スレッド実行状態を各プロセッサが得ることが可能となる。

【００３７】図4は、図1に示したプロセッサシステムの第1の実施例におけるスレッド管理手段についてその第4の実施例を示したブロック構成図である。また、図5は図4に対応して、第1の実施例におけるプロセッサに関してその第2の実施例を示したブロック構成図である。これらの実施例は本発明の請求項6によるプロセッサの制御方法に対応するものである。

12

【００３８】図4において本実施例のスレッド管理手段４２１は図3に示した実施例におけるスレッド管理手段３０２に、フォーク可否決定装置４０１、負荷状態基準装置４０２を付加し、更に図2に示した実施例におけるスレッド管理手段２２１内のロック装置２０１を付加したものである。また、図5において、本実施例のプロセッサ５０１は図1に示した実施例におけるプロセッサ１０１にフォーク可否問い合わせ装置５０２を追加し、更に命令パイプライン１０２に逐次化機能を追加して命令パイプライン５０３にしたものである。

【００３９】次に、これらの対応する実施例の動作を順を追って説明する。命令パイプライン５０３においてフォーク命令が実行されると、フォーク可否問い合わせ装置５０２はスレッド管理手段１２１に対してフォーク可否の問い合わせを情報伝達線４０３を介して行なう。スレッド管理手段１２１は、フォーク可否決定装置４０１に対してプロセッサ５０１が指定したフォーク先のプロセッサへのフォークの可否を問い合わせるとともに、負荷状態テーブル１０９の更新を禁止する。フォーク可否決定装置４０１は、負荷状態テーブル１０９の値とフォーク可能な基準が示される負荷状態基準装置４０２の値を比較することによって、フォークの可否を調べ、情報伝達線４０４にその結果を流す。プロセッサ５０１には情報伝達線４０４によってフォークの可否判断結果が伝えられる。フォーク可能な場合にはフォークメッセージ生成装置１０４によってフォークメッセージが生成される。フォーク不可能な場合には、その情報が情報伝達線５０４によって命令パイプライン５０３に伝えられる。この場合、命令上はフォーク命令を実行したことになるが、命令パイプライン５０３上ではサブルーチンコールと同等に扱われ、自動的に逐次処理が行なわれる。これらのことにより、フォークが不可能な場合でも、フォーク命令によって自動的に逐次化可能となるため、命令パイプライン５０３で実行されるプログラムコードは、フォーク前にフォークの可否判断やフォーク失敗時の例外処理について考慮する必要がなくなり、実行効率向上とコードサイズ減少が可能である。このように、本実施例におけるフォーク命令は負荷状態に応じて自動的に逐次化されるという新しい制御方法を備えたフォーク命令である。

【００４０】図6は、本発明によるプロセッサシステムの第2の実施例を示すブロック構成図である。本実施例は請求項7のプロセッサの制御方法に対応する。

【００４１】図6において、本実施例のプロセッサシス

テムは図1に示したプロセッサシステムの第1の実施例に準じているが、第1の実施例におけるスレッド管理手段121に、図2のロック機構201、図3の加算器301、減算器302、選択装置303を加え、更にフォーク命令によって新たに生成されるスレッドの生成先プロセッサ番号を決定する機構を加えてスレッド管理手段621としている。なお、生成先プロセッサ番号を決定する機構は、フォーク先決定装置601とフォーク命令調停装置602から構成される。なお、本実施例におけるフォーク命令は、自動逐次化を行なうフォーク命令でも行なわないフォーク命令でもどちらでも良い。

【0042】次に、本実施例の動作を順を追って説明する。プロセッサ101の命令パイプライン105においてフォーク命令が実行されると、フォーク先を指定しないフォークメッセージをフォークメッセージ生成装置104で生成し、スレッド管理機構621に情報伝達線605を用いて送る。フォークメッセージは命令調停装置602で調停が行なわれ、フォーク先決定装置601に対してフォーク先決定要求を行なう。これとともに、フォーク先決定を行なっている間、負荷状態テーブル109の更新を禁止する。フォーク先決定装置601は、各プロセッサの負荷状態カウンタ108の値を比較することによってフォーク先のプロセッサ番号を決定する。この結果を指定されたプロセッサ101のフォークメッセージ記憶装置103に情報伝達線606を介して送るとともに、該当するプロセッサの負荷状態カウンタ108の値を加算する。これらの機構により、プロセッサはスレッド生成先の指定を行なうことなく、負荷分散を考慮したスレッドフォーク命令を実行することが可能になる。

【0043】図7は、本発明におけるプロセッサシステムの第3の実施例を示すブロック構成図である。本実施例は請求項8のプロセッサの制御方法に対応する。図7において、本実施例のプロセッサシステムは図1に示したプロセッサシステムの第1の実施例に準じているが、第1の実施例におけるスレッド管理手段121に、図2のロック機構201、図3の加算器301、減算器302、選択装置303を加え、更にスレッド確保命令によってスレッドをあらかじめ確保する機構を加えてスレッド管理手段721としている。なお、スレッドをあらかじめ確保する機構は、スレッド確保判断装置701とスレッド確保命令調停装置702から構成される。

【0044】次に、本実施例の動作を順を追って説明する。プロセッサ731の命令パイプライン105においてスレッド確保命令が実行されると、スレッド管理機構721にスレッド確保要求を情報線703を用いて送る。スレッド確保要求メッセージは命令調停装置702で調停が行なわれ、スレッド確保判断装置701に対してスレッドが確保できるか問い合わせる。これとともに、問い合わせを行っている間、負荷情報テーブル109の更新を禁止する。スレッド確保判断装置701は、各プロセッサの負荷状態カウンタ108の値を比較することによってスレッド確保が可能か判断し、結果を情報源704に流すとともに、確保できる場合には該当するプロセッサの負荷情報カウンタ108の値を更新する。これにより、フォーク命令を実行する前にあらかじめスレッドを確保できるので、最適なフォークが可能となる。

【0045】

【発明の効果】本発明によるプロセッサシステムにおいては、図1の実施例に示したように、複数のプロセッサ間でスレッド管理手段を共有している。このスレッド管理手段は各プロセッサの負荷状態を監視する負荷状態カウンタを有しており、プロセッサがこの負荷状態カウンタをアクセスすることにより、以下のように効率的に動的な並列性のコントロールを行なうことができる。

【0046】・カウンタロック命令とカウンタアンロック命令により、各プロセッサによる負荷状態カウンタの排他的なアクセスが可能になる。すなわち、カウンタロック命令によるあるプロセッサが負荷状態カウンタをロックした後これを参照し、負荷状態カウンタの値によって動的にフォークを行なうかどうかを決定するという動作が可能になる。フォークを行なわなかった場合は、カウンタアンロック命令により負荷状態カウンタのロックを解除する。このようにしてソフトウェア制御によって負荷状態に応じた並列性抽出を行なうことが可能となる。

【0047】・あるプロセッサに対応する負荷状態カウンタの値を、このプロセッサにスレッドがフォークされた際にこれを増加させ、このプロセッサがスレッドの実行を終了する際に減ずるようにすることで、簡単に負荷状態カウンタが動的な負荷状態を表すようにすることができる。

【0048】・より高度なフォーク命令の機能として、フォーク命令が発行された際にスレッド管理手段を参照し、フォークが可能かどうかを負荷状態カウンタの値から判断し、フォークが不可能な場合は当該スレッドを自動的に逐次実行することにより、マルチスレッドによる並列処理をより高性能化することが可能となる。このようなフォーク命令による動作は、ソフトウェア制御ではなく、ハードウェア制御により負荷状態に応じた動的な並列性制御を行なっていることにあたる。

【0049】・従来技術のフォーク命令は、スレッドをフォークする相手先のプロセッサを指定していた。本発明のスレッド管理手段を用いることにより、負荷状態の軽いプロセッサにスレッドをフォークするようにフォーク命令の制御方法を変更し、より柔軟な並列処理が可能となる。

【0050】・あるスレッドをフォークする前に、そのスレッドのフォークが可能かどうかあらかじめ知ってお

15

いた方がプログラム実行上の効率が良くなる場合がある。スレッド確保命令によりスレッド管理手段をアクセスし、スレッドの実行が可能かどうか確認し、可能な場合はあらかじめ負荷状態カウンタの値を増加させ、その後に実際のフォーク命令を実行することにより、このような場合についても効率的なプログラミングが可能となる。なお、スレッド確保命令の後に用いられるフォーク命令については負荷状態カウンタの値を増加させる機能は必要ない。

【００５１】以上のように本発明によるプロセッサ及びプロセッサシステムにより、簡便なハードウェアで動的な並列性のコントロールを行ない、スレッドを用いた並列処理を効率化することができる。

【図面の簡単な説明】

【図1】本発明によるプロセッサ及びスレッド管理手段を含むプロセッサシステムの第1の実施例を示すブロック構成図である。

【図2】スレッド管理手段の第2の実施例を示すブロック構成図である。

【図3】スレッド管理手段の第3の実施例を示すブロック構成図である。

【図4】スレッド管理手段の第4の実施例を示すブロック構成図である。

【図5】プロセッサの第2の実施例を示すブロック構成図である。

【図6】プロセッサシステムの第2の実施例を示すブロック構成図である。

【図7】プロセッサシステムの第3の実施例を示すブロック構成図である。

【図8】従来技術のマルチスレッドアーキテクチャに基づくプロセッサ及びプロセッサシステムの例を示すブロック構成図である。

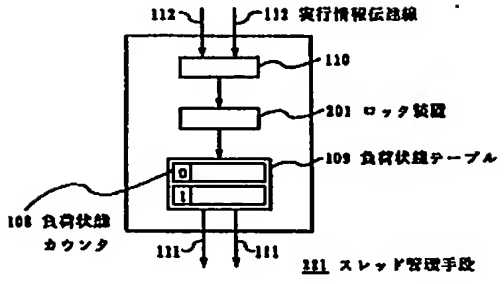
【符号の説明】

101　プロセッサ
103　フォークメッセージ記憶装置
104　フォークメッセージ生成装置
105　命令パイプライン
106　メッセージ伝達線
107　メッセージ伝達線

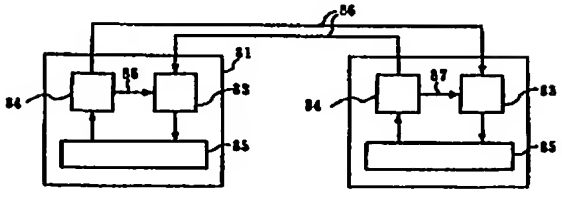
16

108　負荷状態カウンタ
109　負荷状態テーブル
110　メッセージ解読装置
111　負荷情報伝達線
112　実行情報伝達線
121　スレッド管理手段
201　ロック装置
221　スレッド管理手段
301　加算器
302　減算器
303　選択装置
321　スレッド管理手段
401　フォーク可否決定装置
402　負荷状態基準装置
403　情報伝達線
404　情報伝達線
421　スレッド管理手段
501　プロセッサ
502　フォーク可否問い合わせ装置
503　逐次化機能付命令パイプライン
504　情報伝達線
601　フォーク先決定装置
602　フォーク命令調停装置
605　情報伝達線
606　情報伝達線
621　スレッド管理手段
631　プロセッサ
701　スレッド確保判断装置
702　スレッド確保命令調停装置
703　情報伝達線
704　情報伝達線
721　スレッド管理手段
731　プロセッサ
81　プロセッサ
83　フォークメッセージ記憶装置
84　フォークメッセージ生成装置
84　命令パイプライン
86　メッセージ伝達線

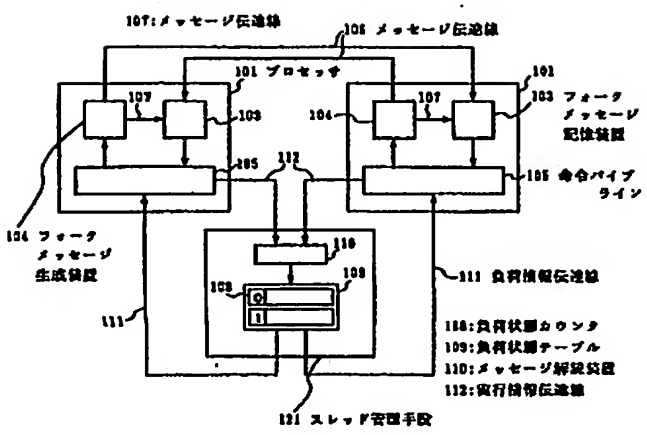
【図2】



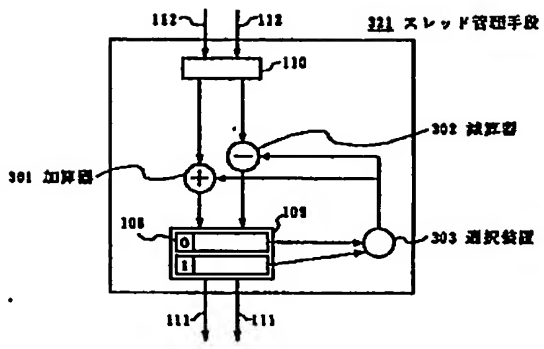
【図8】



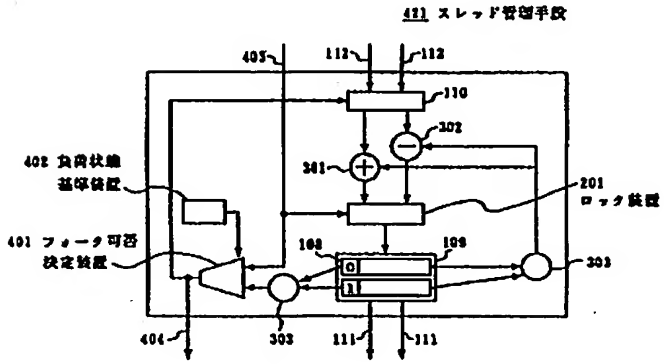
【 図1 】



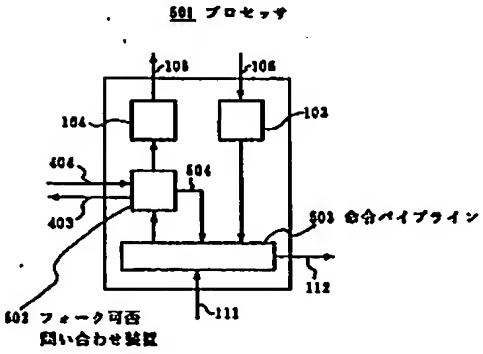
【 図3 】



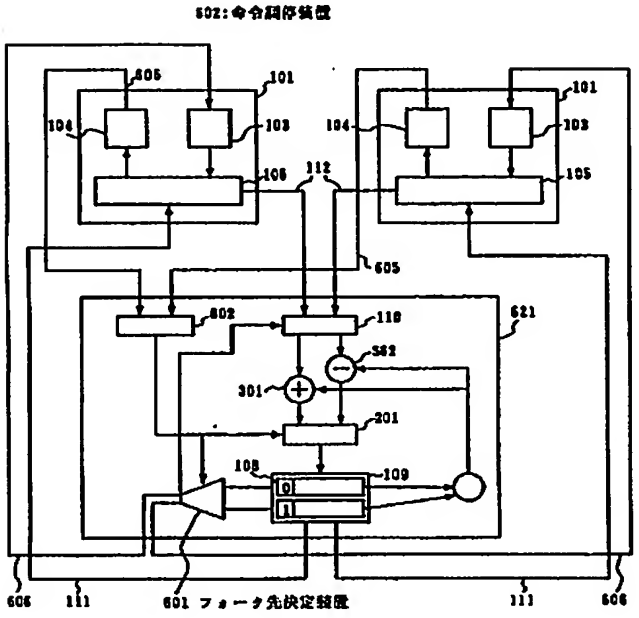
【 図4 】



【 図5 】



【 図6 】



【 図7 】